DETECTION OF FAULTS IN THE TRANSMISSION LINES BY USING 8051 MICROCONTROLLER

A Thesis Submitted in Partial Fulfilment of the requirement for the Degree of Bachelor And Technology in Electrical Engineering

Department of Electrical Engineering

National Institute of Technology Rourkela

By

SUBHAM SWAGAT PATEL (111EE0224)

SANAT KUMAR SETHY (11EE0233)

SATYAJEET LAL (111EE0448)

Under The Supervision of

Dr. SUPRATIM GUPTA

Department of Electrical Engineering,

NIT Rourkela,

Odisha, India - 769008

Certificate

This is to certify that the work in this project entitled "**Development of microcontroller based over-current relay controls**" by **Subham Swagat Patel(111EE0224),Sanat Kumar Sethy(111ee0233) and Satyajit Lal(111ee0448)** has been carried out under my supervision in partial fulfillment of the requirements for the degree of Bachelor in Technology during session 2014-15 in the department of Electrical Engineering, National Institute of Technology Rourkela, and this work has not been submitted elsewhere for a degree.

The candidates have fulfilled all the prescribed requirements.

In my opinion, bachelor of technology degree in Electrical Engineering may be awarded based on this thesis.

Dept. of Electrical Engineering

National institute of Technology **Prof. Supratim Gupta**

Rourkela-769008

Place: Rourkela

ACKNOWLEDGEMENTS

We would like to express our gratefulness to our project supervisor **Dr. SUPRATIM GUPTA** who has always been there to guide us in carrying out this project. He has always supported us with his educative guidance and his readiness to help us at any times has been a great source of motivation for us. We would also like to give our sincere thanks to **Mr. Bhanupratap Behera** and all other faculty members who have been a great hand of help without whom it would have been difficult to complete our project work. Also articles, journals and books from online and offline sources helped us a lot in carrying out our project work. At last it would have never been possible to concentrate and work on this project without the blessings of our parents and the Almighty.

Subham Swagat Patel (111ee0224)

Sanat Kumar Sethy (111ee0233)

Satyajeet Lal (111ee0448)

ABSTRACT

Electricity has become the most sought after amenity for all of us. Gone are the days when electricity would be only limited to cities. It is now reaching to every distant parts of the world. So we have now a complex network of power system. This power is being carried by the transmission lines. These lines travel very long distances so while carrying power, fault occurring is natural. These faults damages many vital electrical equipments like transformer, generator, transmission lines. For the uninterrupted power supply we need to prevent these faults as much as possible. So we need to detect faults within the shortest possible time. Microprocessors and microcontroller based systems used for these fault detection have been advancing rapidly. The proposed paper simulates Numerical Overcurrent relay that detects faults using microcontroller and ADC. These relays are more reliable and have faster response than the traditional electromechanical relays and Static relays. They have increased range of setting, high accuracy, reduced size, and lower costs, along with many other functions, such as fault event recording, auto-resetting, etc.

This project is about designing the Numerical relay where the fault is detected when the input value exceeds the reference value set in the relay which then gives the trip signal to the circuit breaker.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Relay acts as an electrical switch that is operated by a circuit of small power rating to control circuit of larger power rating. Electromechanical Relay operates on electromagnetic principle. It has a magnetic coil which is energized by electric current to behave as a magnet. These relays which prevents faults are called as Protective relays. Nowadays microcontroller based relays are gaining more popularity than the traditional Electromechanical relays and used extensively to prevent faults due to its faster response, reliability, less cost, compact size etc. Overcurrent relay act on the principle that when the input current or voltage value exceeds the predefined set value then the relay works and sends a trip signal to the circuit breaker.

This Project will be controlling the value of relay pickup current by the help of Microcontroller. We have used P89V51RD2 8051 microcontroller, ADC 0808, LCD JHD162A (16x2 display) to detect faults. Fault is detected and Trip signal is generated when the input current value is greater than that of relay preset value.

FAULTS

It is an abnormal condition caused by many factors related to nature like lightning, wind, natural disaster & human error. It means there is a flow of very high value of current above the normal value. Due to this many electrical apparatus like transformer, generator, and transmission lines get affected because of overheating and insulation failure.

TYPES

1) Shunt fault or short circuit fault which is classified into unbalanced (asymmetrical) and balanced (symmetrical) faults like L-G, L-L, L-L-G and L-L-L, L-L-L-G respectively.

These occur due to insulation failures and falling of tree branches. These types of faults involve ground.

2) Series or open circuit fault which are due to melting of the conductor because of overloading or breakage of conductor due to wind effect.

ELECTROMECHANICAL RELAYS

Electromechanical protective relays are operated by magnetic induction. In this an electromagnet is formed by a coil of wire wound around an iron core which will move an armature that is connected to the switch of the controlled circuit. If a relay is not energized its armature wont control the switch. When current exceeds the set current in the coil the armature will move and control the switch as long as it is energized.

OVERCURRENT INDUCTION DISC TYPE RELAY

It works by inducing currents in a disk that is free to rotate which will operate a contact. Induction relays require AC. If we are using two or more coils they should be at same frequency otherwise net operating force is not produced.

STATIC

It uses electronic amplifiers like vacuum tube amplifiers. It has no moving mechanical parts unlike in Electromechanical relay. It uses analogue electronic devices instead of magnetic coils and mechanical parts to obtain the relay characteristics.

NUMERICAL RELAYS

Numerical relays are microprocessor and microcontroller based relays having its own memory Numeric relays take the input analog quantities and convert them to numeric values. Electromechanical and Static relays are not multifunctional unlike Numerical relay.

DISADVANTAGES OF ELECTROMECHANICAL RELAYS

1) Electromechanical Relay uses mechanical parts that makes it bulky and larger in size. Flag system is used to tell whether the relay is activated or not.
2) It is not flexible as we can not modify its characteristics and functional operations unlike in software supported Numerical relay.

3) It is not as reliable as the numerical relay.
4) It does not provide multifunctional operations to control various features related to fault.
5) Auto-resetting is not possible in these relay.
6) It does not have memory to record fault related data.

ADVANTAGES OF NUMERICAL RELAYS

1) Compact Size: Numerical relay is compact in size, and uses LCD to indicate relay activation. It requires less wiring so it is not having complex architecture.
2) Flexibility: We can modify its functional operation by changing codes in software.
3) Reliability: It is more reliable because of less interwiring, use of less components and reduced component failures.
4) Multi Functional Capability: Displaying results and data in LCD, recording fault related data etc. makes it a multi-functional in its operation.
5) Different Types of Relay Characteristics: We can get Definite Time Characteristics of different time values and Indefinite Time Characteristics of various values from it as they are stored in the microcontroller memory.
6) Digital Communication Capabilities: It is easily interfaced with different digital equipments.
7) Low burden: It has less burden on Instrument transformer.
8) Sensitivity: It has high sensitivity and pickup ratio.
9) Speed: It has the highest speed of operation among other relays.
10) Data History: It has memory of its own so it can record the various details of faults like nature, magnitude and duration of fault.
11) Auto Resetting and Self Diagnosis: It can decide whether normal condition has arrived after the fault.

1.1 OBJECTIVE: To devise and program an 8051 microcontroller based Numerical relay using Assembly Language to detect fault in Transmission lines.

CHAPTER 2

BACKGROUND AND LITERATURE AND COMPONENTS DESCRIPTION

Here we will discuss about the importance of carrying out this project. We will also discuss about different components used in our project.

2. 1 NEED FOR PROTECTION

Fault introduces serious danger on both electrical apparatus and people. Therefore we have to protect ourselves as well as the equipments from these faults. Without it power system will fail in no time.

Various issues need to be protected are:

- Safety for People
- Equipment safety: Keeping equipments safe from various electrical abnormal and faulty conditions.
- Power system stability: Maintaining a continuous and reliable power supply.

2. 2 OVERCURRENT RELAY

Over current is defined as any current which is more than the rated current rating of the equipment or a conductor. This may be caused by overload, short circuit, or ground fault. When current flows through a conductor it produces heat. So at faulty condition large current results in overheating which may damage equipments. So in order to save them from overheating and damage overcurrent relays are used.

The current from the transmission lines goes to the overcurrent relay through the current transformer in the form of AC. In normal condition relay remains in open state and in closed state in overcurrent fault situation. This relay has two basic settings which are called Time Setting Multiplier and Plug Setting Multiplier. Time setting multiplier adjusts the travelling distance of moving contact that is one of the causes of operational time delay of the relay when fault occurs. Another cause is speed of moving contact which depends upon the level of fault current. The pickup current value is decided by Plug setting multiplier.

Types

1) Instantaneous Overcurrent Relay- This type of relay operates at an instant without any delay when input current exceeds the set value of current. It operates in definite time. These are used mostly on outgoing feeders.
2) Definite Time Overcurrent Relay- This type of relay operates when two conditions are met i. e. input current exceeding the relay setting value and the desired time delay is reached. The operation of these relays are independent of the fault current level rather the time delay provided to it to work.
3) Indefinite Time Overcurrent Relay- In this type of relay the operating time is inversely proportional to the magnitude of fault current. So it will operate faster for high current and slower for low current values. Again it has 3 variants according to different time vs current characteristics.

   A) Normal Inverse Time Overcurrent Relay- In this there is a small change in time per unit of change of current occurs. Operating Time $T = 0.14x\ (TMS)/ ((PSM)^{0.02}-1)$.

   B) Very Inverse Time Overcurrent Relay- In this there is comparatively a larger change in time per unit of change of currents than the Normal inverse time relay. Operating Time $T= 13.5 x\ (TMS) / (PSM-1)$.

   C) Extremely Inverse Time Overcurrent Relay- It has more inverse character than the Very Inverse Time Relay. Operating Time $T= 80 x\ (TMS) / (PSM2-1)$.

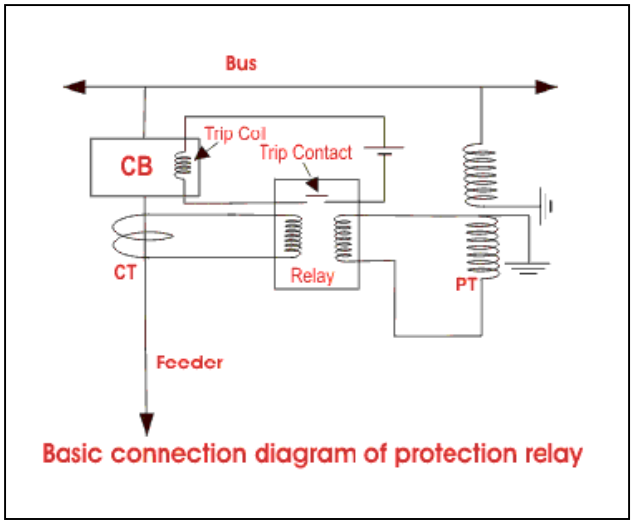


Fig. 1. Simple Relay Circuit [8]

2. 3 P89V51RD2 MICROCONTROLLER

The P89V51RD2 is a microcontroller of 8051 family which is built using CMOS technology, hence it is related to 80C51 microcontroller. A main feature of it is its X2 mode option. We can choose to run any operation either in conventional 80C51 clock rate which is 12 clocks per machine cycle or select the X2 mode which is 6 clocks per machine cycle. This will help execute twice one-cycle instructions per second or twice 500,000 two-cycle instructions per second than in 80C51 conventional clock rate. Also with the help of this feature we can reduce the EMI by keeping the same performance by reducing the clock frequency to half. Both parallel programming and in serial ISP are supported by its Flash Memory. Gang-programming is done through Parallel programming at high speed thus reducing programming costs and time while ISP (In-System Programming) allows any operation to be reprogrammed in the end product by modifying the codes in the programming software. The capability to update the application code helps in having a wide range of applications possible. It is also IAP (In-Application Programmable) that allows its

Flash memory to be reconfigured even during running of the application.

FEATURES

- 5 V Operating voltage from 0 to 40 MHz
- 64 kB of on-chip Flash program memory with ISP
- Supports 12-clock (default) or 6-clock mode selection via software or ISP
- 64 kB Flash and 1024 bytes of data RAM.
- 80C51 CPU
- IAP
- Enhanced UART SPI (Serial Peripheral Interface) that allows it to interface easily with other peripheral devices through serial communication.
- Four 8-bit I/O ports with three high-current Port 1 pins (16 mA each)
- Three 16-bit timers/counters
- Eight interrupt sources with four priority levels
- Low EMI mode.
- Programmable Watchdog timer (WDT) - It protects the application against software deadlock and automatic recovery. It should be refreshed within a given period of time to avoid the deadlock otherwise internal hardware reset will be enabled.
- Second DPTR register- The device has two 16-bit data pointers DPTR0 and DPTR1
- TTL- and CMOS-compatible logic levels
- Brown-out detection- This causes the microcontroller to reset to save it from supplied voltage VDD fluctuations. Its threshold value for P89V51RD2 is 3. 85 V. It is triggered when the voltage value goes below 3. 85 V.
- Low power modes

  Power-down mode with external interrupt wake-up

  Idle mode

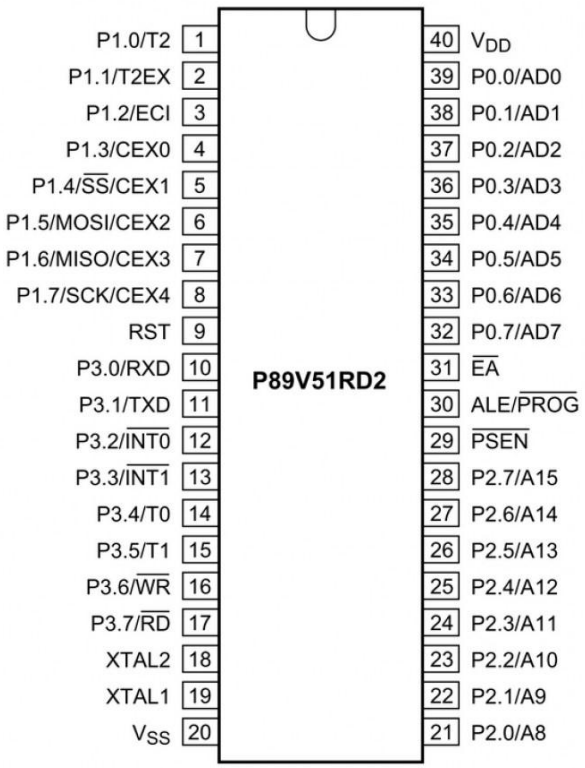


Fig. 2. P89V51RD2 Pin-out Diagram [3]

2. 4 ADC 0808 and JHD162A LCD

ADC 0808

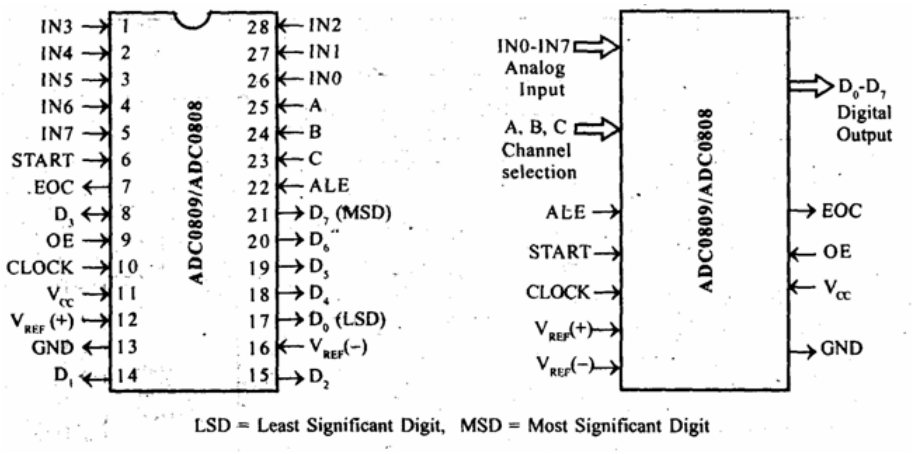


Fig. 3. ADC 0808 Pin-out Diagram [3]

ADC (Analog to digital converters) are used to convert analog signals to digital signals. In our physical world, mostly used signals are analog. Microcontroller can only operate with digital signal, so we need ADC. Earlier ADC 0804 were used, but since it can only operate with one analog signal at a time, and having less memory so, ADC 0808 was introduced which can operate with 8 signals at a time.

PIN Configuration

It has 28 pins. Vref (+) and Vref (-) are used to set a base voltage. Vref (+) is usually given 5 V and Vref (-) is used as ground. A, B and C are used for selecting the channels i. e. IN0-IN7. ALE, EOC, OE, SC are used to fetch data from ADC 0808. We use Vcc pin to supply 5V to ADC and there is a GND pin which we use to ground the ADC. To get data from ADC following steps are used-

1. An analog signal is selected by providing different values to A,B and C
2. We give an L to H signal to ALE to activate ALE.
3. Then, EOC is checked. An H to L output from EOC indicates that ADC has completed its process of conversion and data is ready to be picked up.
4. To read data OE pin is given an L to H pulse.

FEATURES

- We can easily interface it with any microcontroller.
- We do not need to adjust its zero or full scale value.
- It uses a multiplexer having 8 channels with address logic.
- Its input range is from 0V to VCC.
- ADC 0808 is similar to MM74C949.

JHD162A LCD

Fig. 4. LCD 16X2 Pin-out Diagram

Nowadays LCDs are gaining popularity and are replacing LEDS. LCD modules are available in different version like 1x16 (one line -16 characters) or 2x16 (two lines -16 characters) or 2 x 20(two lines 20 characters) etc.The one we used is having 2x16 display. It has total 14 pins. They have advantages like-

- Low cost
- It can display numbers, characters and graphics whereas LEDs can only display numbers and few characters.
- Programming for characters and graphics is simple

Table. 1. PIN Configuration of LCD

PIN NO	FUNCTION	NAME
1	Ground(0V)	Ground
2	Supply Voltage (5V)	Vcc
3	Contrast Adjustment through a Potentiometer	Vee
4	Selects Command Register when Low,and Data Register when High	Register Select
5	Low to Write to a Register,High to Read from the Register	Read/Write
6	Sends Data to Data Pins when a High to Low Pulse is Given	Enable
7	8-Bit Data Pins	D0
8		D1
9		D2
10		D3
11		D4
12		D5
13		D6
14		D7

- **Vcc , Vss, Vee** – Vcc and Vss are used to provide 5V and ground, respectively and V_{ee} is given 5V through a rheostat to control the contrast of the LCD
- **RS** – If RS=0, it allows the user to give command to LCD like clear screen, return home, shift display etc
- **R/W** – If R/W=0, we can write information to LCD and if R/W=1 information is read.
- **E, Enable** – When data is available in pins, a high to low pulse is given to this pin to latch information.
- **D0-D7 –** These are 8 data pins of LCD.

CHAPTER 3

METHODOLOGY

Here we will be explaining about the whole steps followed in the operation of the relay. We will also be explaining about the algorithm followed in finding out the RMS value of the current. To sum up we have posted the flowchart of the operation of the whole project and a picture taken of our hardware model.

3. 1 WORK DONE AND STRUCTURAL ANALYSIS

We are doing the model with 5V supply condition means all the devices i. e. ADC, Microcontroller, LCD are working on 5V supply. We have given also 5V as the voltage reference to the ADC. We have fed a continuous analog sinusoidal voltage signal to the ADC channel number 0 by selecting the values of A=0, B=0, C=0. We have given clock converting frequency to ADC 0808 as 691. 1875 KHz by the help of two D-flip-flops IC 74LS74 each containing two D-flip-flops. ADC performs sampling, quantization and encoding of the analog signal thus producing digital binary data. This binary output digital data is obtained from the 8 output pins of the ADC. The output from the ADC is fed into the port 1 of the microcontroller. The output of the microcontroller is fed into the LCD from the port 0 through external pull-up resistor pack of 8 resistors each of 10KΩ. Output from the ADC is represented as:

$I = 2^n \times (Vin / Vref)$, where Vref is the refence voltage 5V

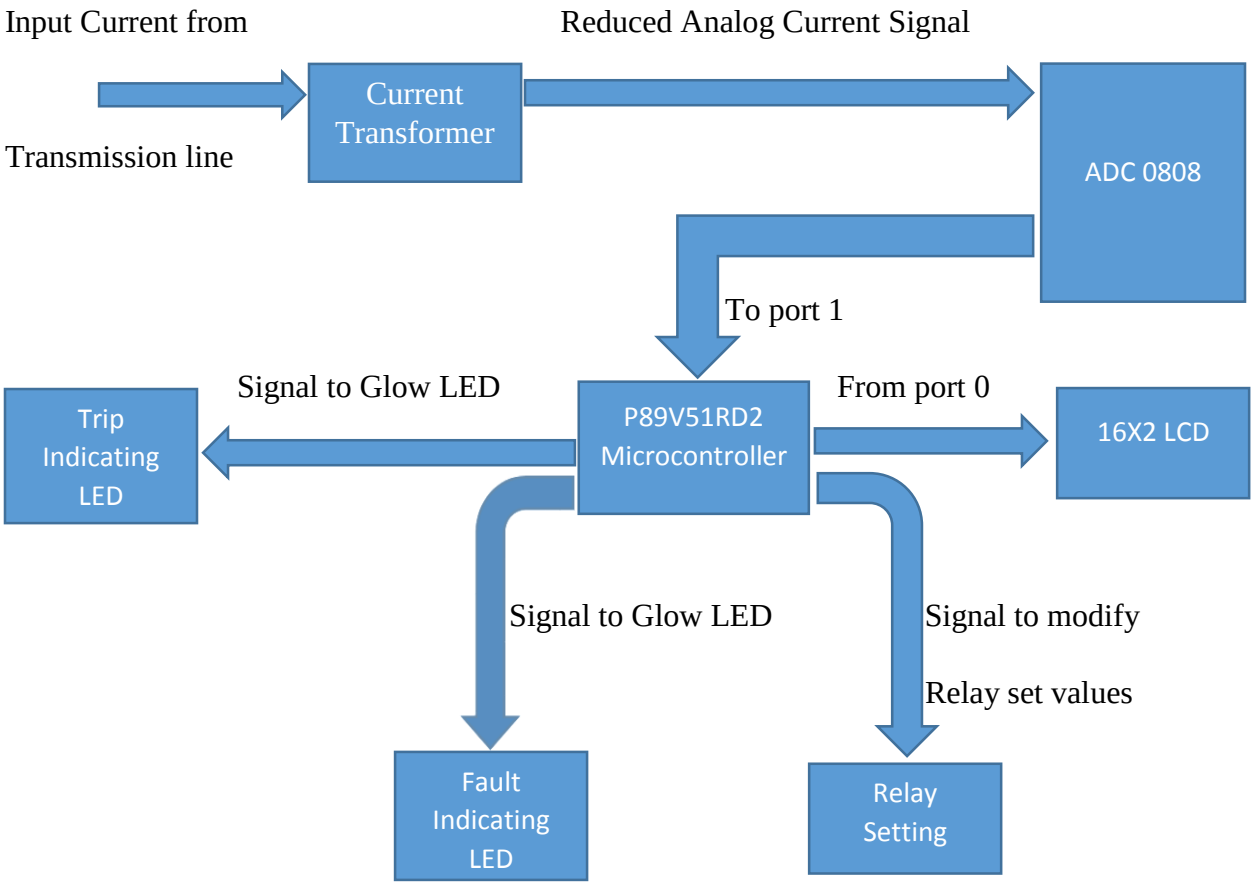
Vin is the input voltage

n is the number of bits (here 8)

STEPS

1) Analog signal (current) is obtained from the transmission lines and is reduced by the current transformer.
2) Then the current signal is transformed to voltage signal by using current sensor circuit.
3) The analog voltage signal is then converted to digital binary signal using ADC.
4) The RMS value of the current is found out and displayed on the LCD.
5) In Microcontroller to which ADC is interfaced, program is written to compare the input current value with the relay preset current value so as to determine the occurrence of fault i. e. if Iin> Iref then fault occurs otherwise no fault.
6) The result is then displayed on the LCD.
7) Facility to change relay preset value, and definite time setting is provided if the operation is invoked.

FIG. 5. BLOCK DIAGRAM OF NUMERICAL OVER CURRENT RELAY



3. 2 RMS Calculation by Division by 7 Method

- STEP1: First, the pick value of the analog signal is found and it is stored in a register in hexadecimal code.
- STEP2: Then, the hexadecimal number is divided by 7 and the quotient is stored in accumulator ie A register and remainder in register B.
- STEP3: Again the quotient in A is divided by 10 and the quotient found is stored in R6 and remainder in R5 of bank 0.
- STEP4: The value in register B is transferred to R2 of bank 0.
- STEP5: Finally, we get 3 numbers, one is stored in R6, the other in R5 and the last one in R2 of bank0. The decimal point is placed after the value stored in R6. So, we get the rms value as **R6 . R5 R2**.

E. g. since ADC is 8 bit and Vref is given as 5V, so the maximum hexadecimal value would be 255 for 5V. Let us assume the pick value of the analog signal found to be 3V. So, by calculation its hexadecimal will be 153 and the actual rms value for 3V pick will be 2. 12.

From our method, first 153 is divided by 7, then we get quotient as 21 and remainder as 6. '21' is stored in register A and '6' in register B. Then, 21 in divided by 10 and its quotient is stored in R6 i. e. '2' and remainder i. e. '1' is stored in R5 of bank 0. The value '6' stored in register B is transferred to R2 of bank 0. Decimal is put after value'2' , the number next to decimal will be the one stored in R5 i. e. '1' and the final value will be the number stored in R2 i.e. '6'. So, we get the rms value as 2.16

ERROR

Actual rms value is 2. 12 and the value we get is 2. 16.

Error= (Measured value – True value) / True value

So, error comes out to be 1. 88%

3. 3 ALGORITHM STEPS FOR NUMERICAL OC RELAY

STEP1: ADC, LCD, and Relay Control pins are defined at first.

STEP2: LCD is initialized and displays "Current:"

STEP3: Maximum Sampled Value is found out by ADC.

STEP4: RMS value of current is found out and displayed on LCD.

STEP5: Comparison between the input current value and relay set value. If carry bit=1 then Microcontroller sends signal fault indication and trip at different definite time setting Values and displays "Fault in Line. If carry bit=0 then LCD displays "No Fault in Line".

STEP6: If fault occurs then check for LCD reset operation.

- If yes then go to STEP2 otherwise continuously displays "Fault in Line "on LCD.

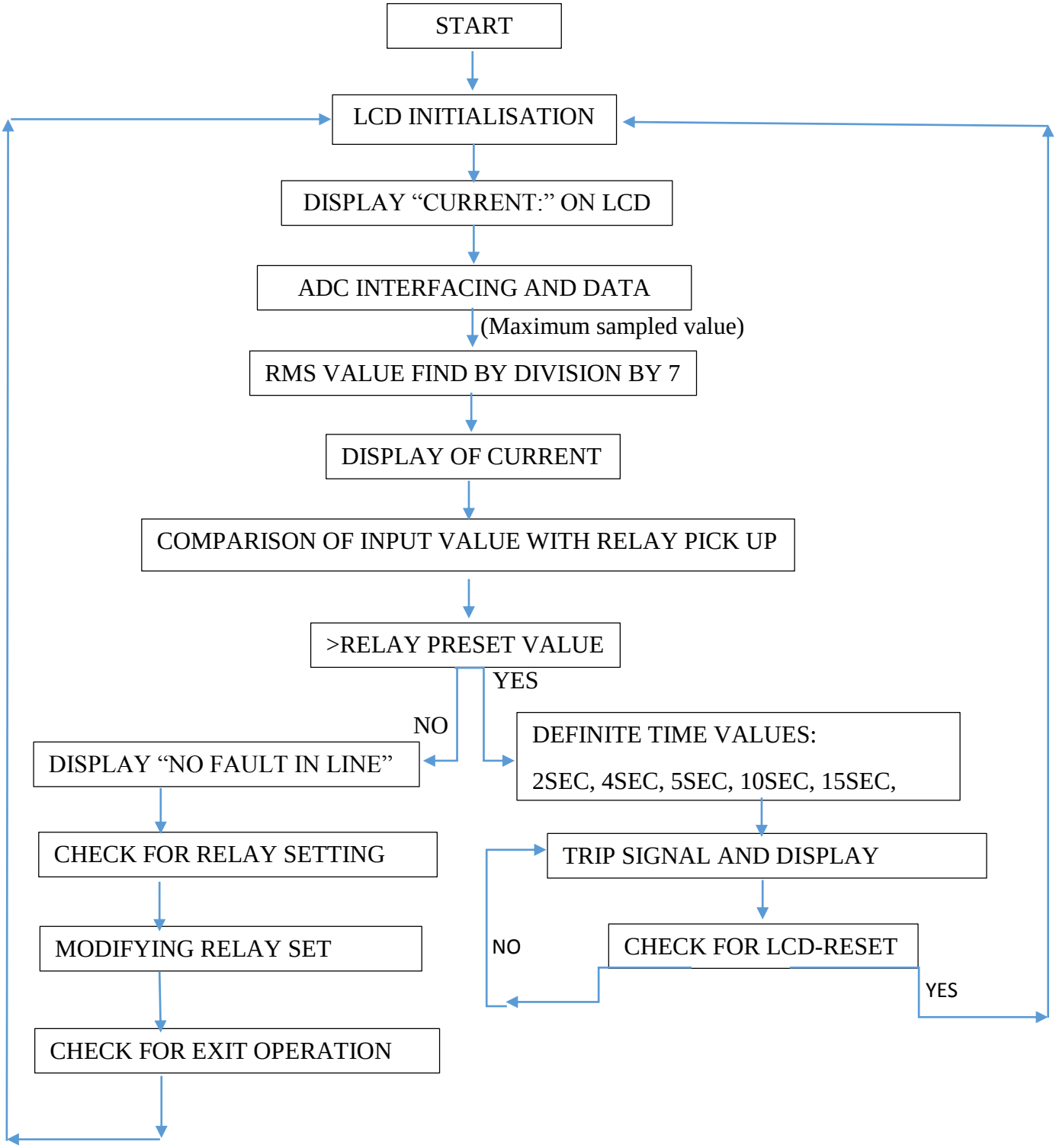
If no fault occurs then check for Relay setting operation.

- If yes then relay set current is modified, definite time for trip is set.

  Meanwhile check for exit operation continuously. If yes then go to STEP2, otherwise continue.

- If no then go to STEP2 and continue.

Fig. 6. FLOWCHART OF PROJECT



3. 4 HARDWARE IMPLEMENTATION

Here we will show different connections between ADC-Microcontroller, Microcontroller-LCD, and Microcontroller-Relay Switches/LEDs etc.

Table. 2. CONNECTION TABLE

Sl. No.	Name of the Port/PIN No.	Connected From	Connected To
1	P0. 0	Microcontroller	LCD pin no. 7 (D0)
2	P0. 1	Microcontroller	LCD pin no. 8 (D1)
3	P0. 2	Microcontroller	LCD pin no. 9 (D2)
4	P0. 3	Microcontroller	LCD pin no. 10 (D3)
5	P0. 4	Microcontroller	LCD pin no. 11 (D4)
6	P0. 5	Microcontroller	LCD pin no. 12 (D5)
7	P0. 6	Microcontroller	LCD pin no. 13 (D6)
8	P0. 7	Microcontroller	LCD pin no. 14 (D7)
9	P1. 0	Microcontroller	ADC pin no. 21 (D1)
10	P1. 1	Microcontroller	ADC pin no. 20 (D2)
11	P1. 2	Microcontroller	ADC pin no. 19 (D3)
12	P1. 3	Microcontroller	ADC pin no. 18 (D4)
13	P1. 4	Microcontroller	ADC pin no. 8 (D5)
14	P1. 5	Microcontroller	ADC pin no. 15 (D6)
15	P1. 6	Microcontroller	ADC pin no. 14 (D7)
16	P1. 7	Microcontroller	ADC pin no. 17 (D8)
17	P2. 0	Microcontroller	ADC pin no. 6 (ADC_SC)
18	P2. 1	Microcontroller	ADC pin no. 7 (ADC_EOC)
19	P2. 2	Microcontroller	ADC pin no. 25 (ADC_A)
20	P2. 3	Microcontroller	ADC pin no. 24 (ADC_B)
21	P2. 4	Microcontroller	ADC pin no. 22 (ALE)
22	P2. 5	Microcontroller	ADC pin no. 23 ((ADC_C)
23	P2. 6	Microcontroller	LCD pin no. 6 (EN)

24	P2. 7	Microcontroller	LCD pin no. 4 (RS)
25	P3. 1	Microcontroller	Trip LED
26	P3. 2	Microcontroller	LCD Reset Switch
27	P3. 3	Microcontroller	Relay Right Switch
28	P3. 4	Microcontroller	Relay Down Switch
29	P3. 5	Microcontroller	Relay Left Switch
30	P3. 6	Microcontroller	Relay Up Switch
31	P3. 7	Microcontroller	Fault Indicator LED
32	IN0	ADC pin no. 26	Potentiometer

3. 5 FUTURE SCOPE FOR MODIFICATION

We have two types of fault which comes under short circuit fault- one is symmetrical and the other is asymmetrical. L-G, L-L, L-L-G comes under symmetrical fault and L-L-L and L-L-L-G comes under symmetrical fault. Our setup is limited to detect fault in a single phase if current exceeds the set value and provides signal to trip. Our set-up in this project can be modified to determine the different types of faults (e. g. L-G, L-L, L-L-L, L-L-L-G) by adding two more similar set-up so that we can have three phases. From those phases we can generate the negative, positive, and zero sequence components of the fault current and then by checking the different conditions like equality or inequality among positive, negative, zero sequence currents we can determine the types of fault.

RESULTS

Simulations are done in Proteus 8. 0 and the programcode is written in KEIL uVision Software Platform.

Fig. 7. Input Current during normal condition

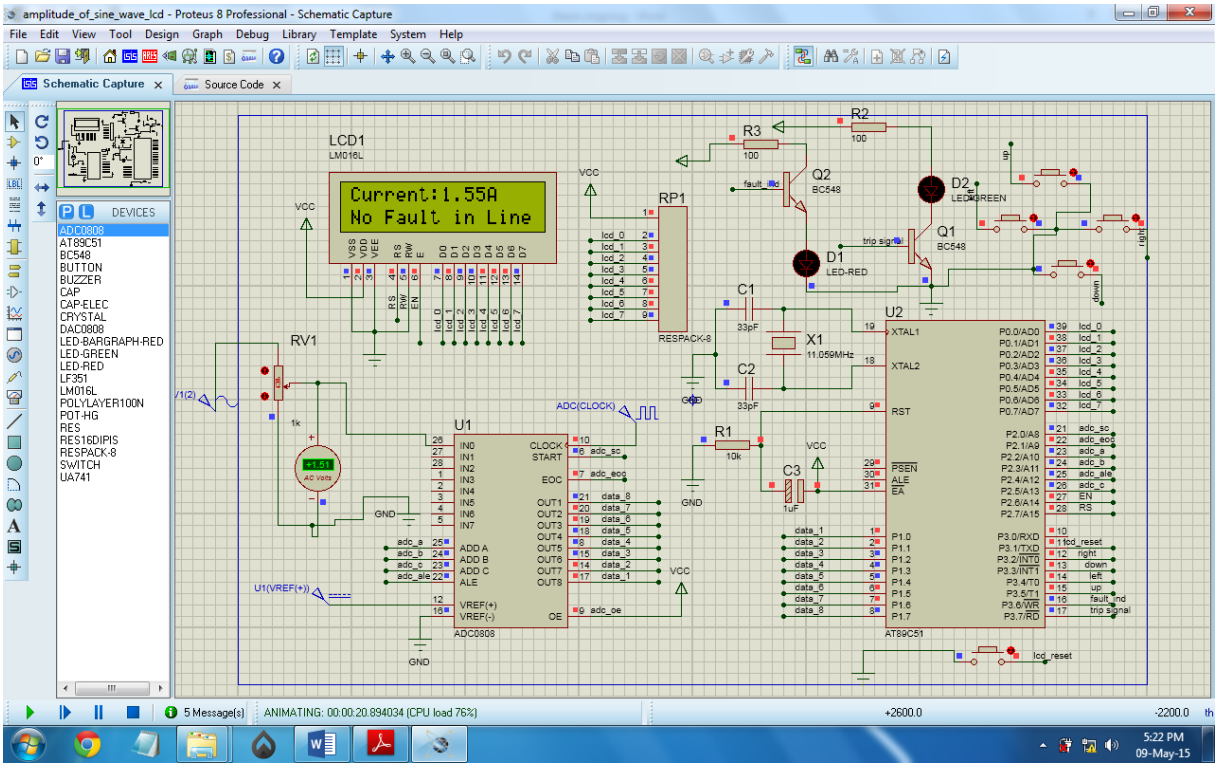


Fig. 8. Relay Set Current-2. 00A (default) and Definite Time Setting Delay-4 Sec (default)

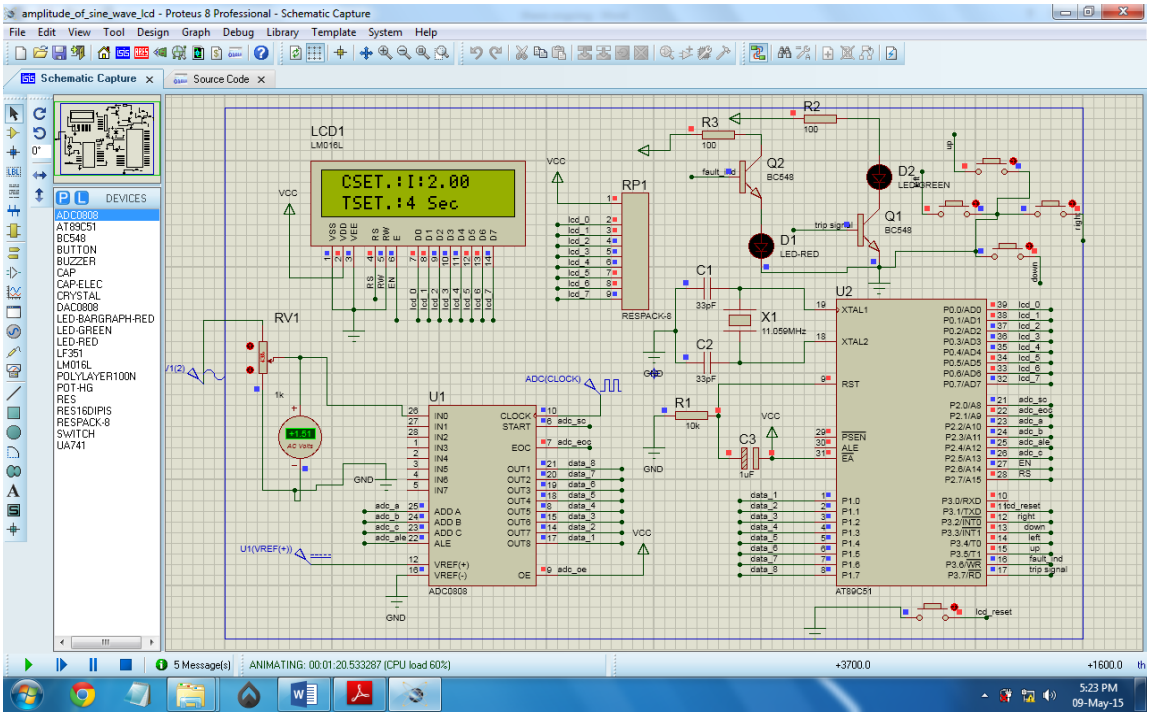


Fig. 9. During Fault, Fault Current=2. 02A and Fault is indicated by Red LED

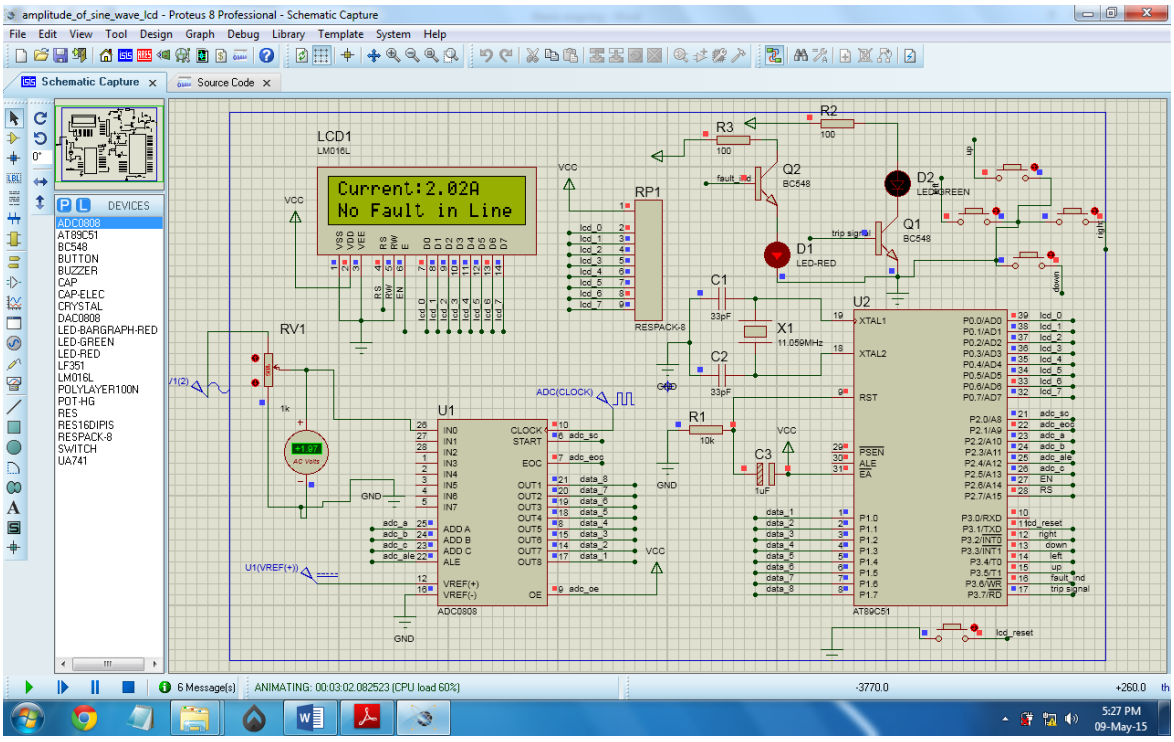


Fig. 10. During Fault, Fault Current=2. 02A and Trip is indicated by Green LED

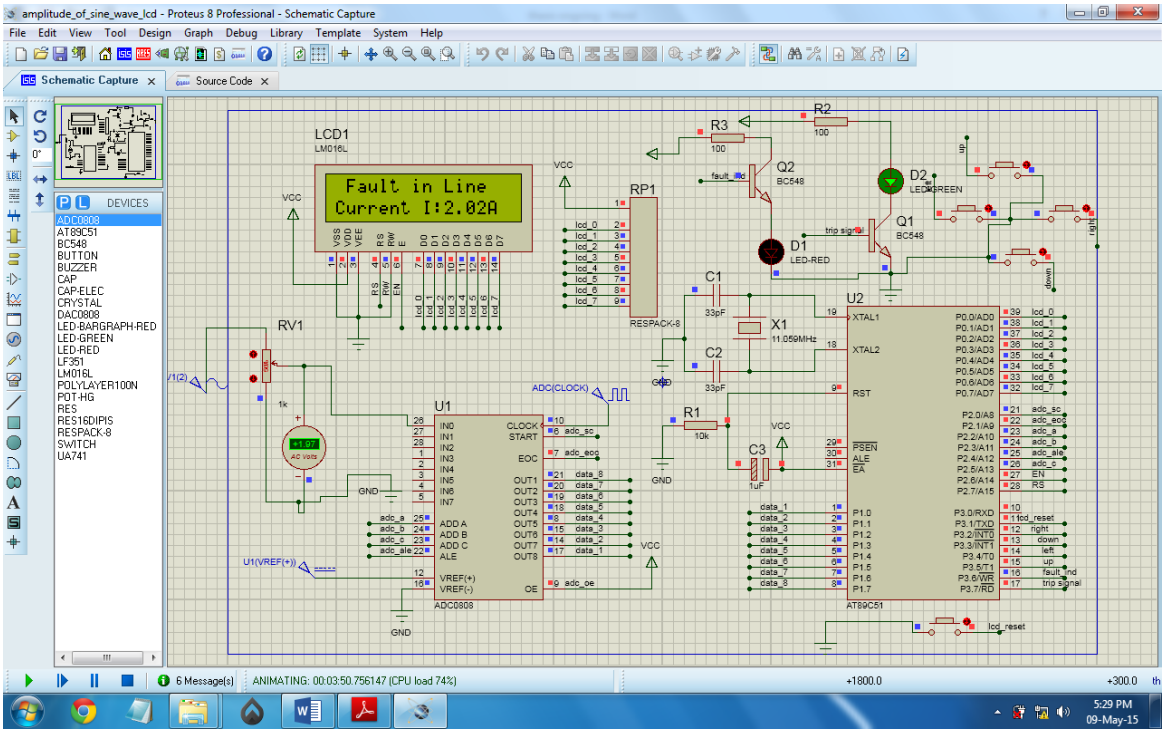


Fig. 11. Setting the Definite Characteristic of the Relay

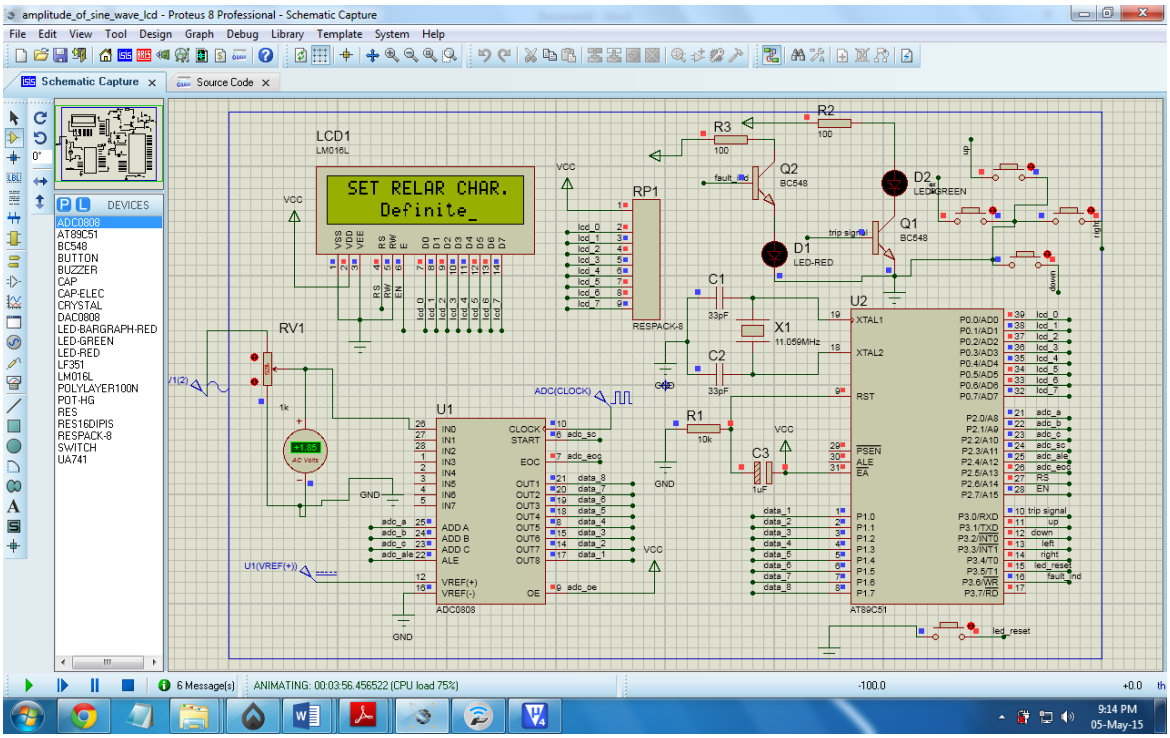


Fig. 12. Setting Definite Time=15sec

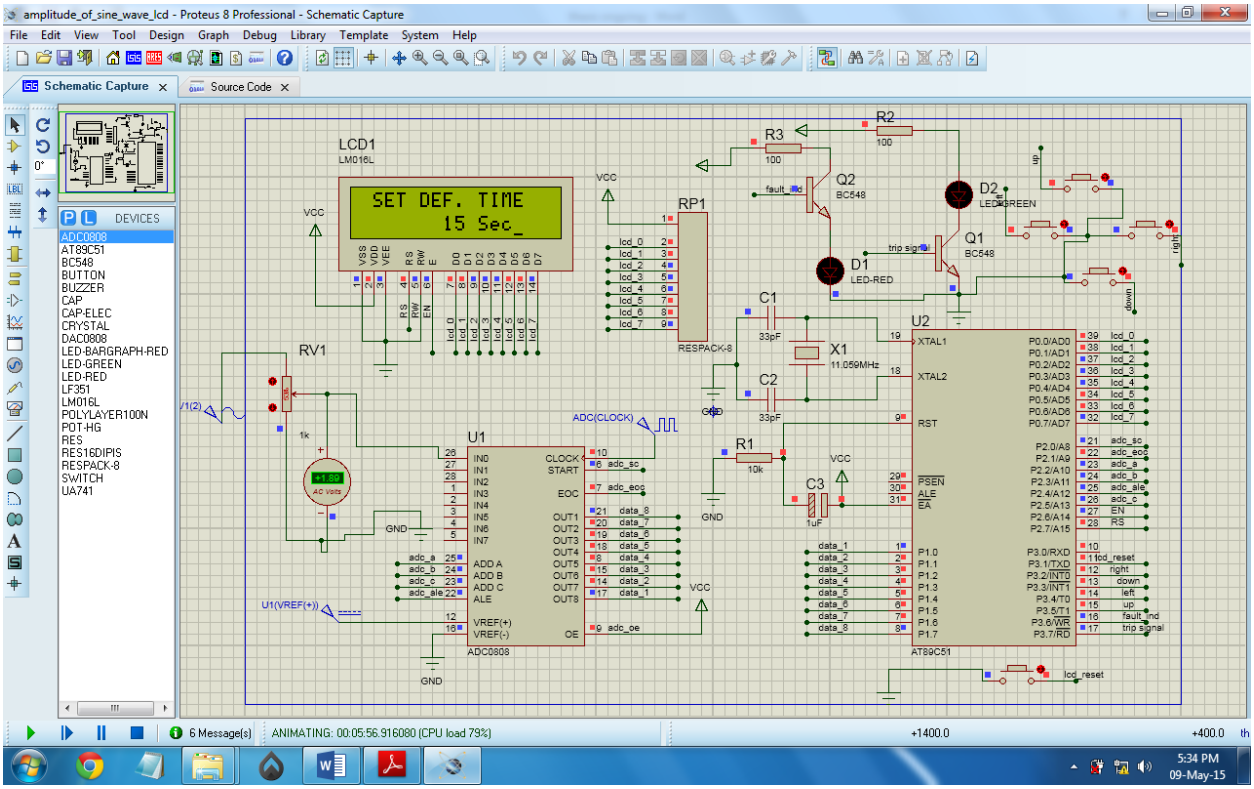
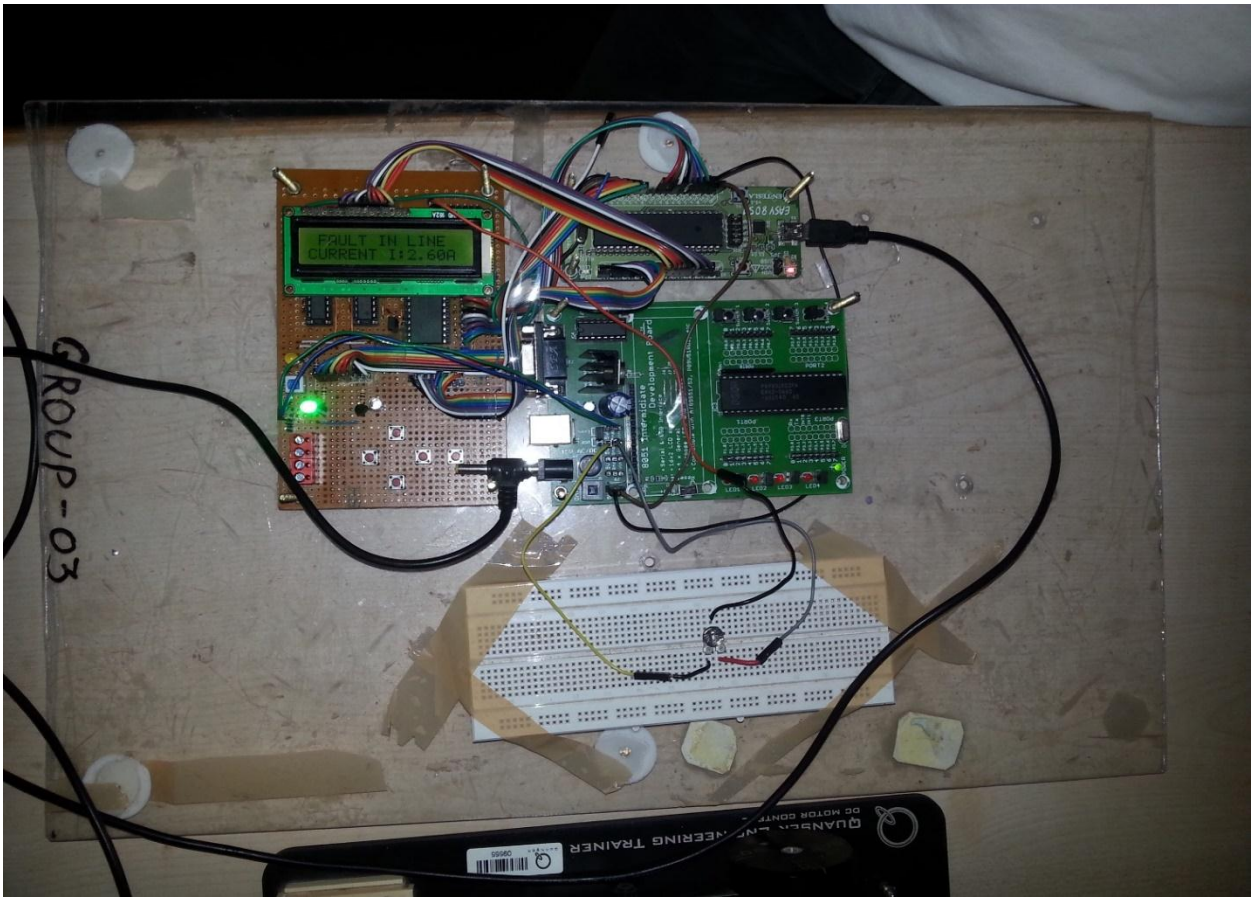


Fig. 13. HARDWARE MODULE



CONCLUSION

We have been able to incorporate successfully the detection of fault by using ADC and Microcontroller by taking both the input voltage as DC and AC Sinusoidal. It is observed that when the current value obtained from the secondary current value of the current transformer is greater than the preset value of the relay then the fault is detected by the microcontroller. The result is displayed on the LCD screen. We can set different trip time delay using definite time characteristics of the relay. By this project it can be ensured faster detection of faults than the electromechanical relay on the power lines and their advanced analysis can be studied from the recorded data by the microcontroller. Also the method we followed to find out the RMS value of the current gave error of about 1. 8-2. 3 % of the actual calculated RMS value.

REFERENCES

[1] Mazidi Muhammad Ali, Mazidi Janice Gillispie, and Mckinlay Rolin D., The 8051 Microcontroller and Embedded Systems using assembly and C.

[2] Gray Nicholas, ABCs of ADC Analog to Digital Converter Basics, National Semiconductor.

[3] ADC 0808 and P89V51RD2 Datasheets , National Semiconductor and Philips Corporation respectively.

[4] Microcontroller based Fault Detector , International Journal of Advancements in Research & Technology, IJOART Technical Research Paper, Volume 1, Issue 5, October-2012 1 ISSN 2278-7763.

[5] Chandra Shekar. P**.,** Transmission Line Fault Detection & Indication through GSM ISSN: 2347 - 2812, Volume-2, Issue -5, 2014.

[6] Kasztenny Bogdan , and RosoIowski Eugeniusz , A Digital Protective Relay as a Real-Time Microprocessor System. O-8186-7889-5197 1997 IEEE.

[7] Harun Zoolnasri Bin Abu , Over Current Protection Relay Using Pic Micro Controller, University Malaysia Pahang, 2007

[8] http://www. electrical4u. com/ & http://en. wikipedia. org/wiki/Intel_MCS-51

APPENDIX

Assembly language codes for whole project

```
;adc address and control pins are defined
ADC_A        BIT    P2. 2
ADC_B        BIT    P2. 3
ADC_C        BIT    P2. 5
ADC_SC       BIT    P2. 0
ADC_ALE  BIT        P2. 4
ADC_EOC  BIT        P2. 1
TRIP_SIGNAL BIT P3. 7
FAULT_IND BIT P3. 6
UP BIT P3. 5
DOWN BIT P3. 3
LEFT BIT P3. 4
RIGHT BIT P3. 2
LED_RESET BIT P3. 1

;lcd control pins are defined
RS      BIT    P2. 7
EN      BIT    P2. 6

 ;program starts from this address
                              ORG  0000H
                              SETB PSW. 4
                              MOV R3,#2D
                              MOV R5,#0D
                              SETB PSW. 3
                              MOV R2,#2D
                              MOV R5,#0D      ;definite time setting
                              MOV R3,#0D      ;value for displaying relay char
```

```
                    MOV R4,#5D;
                    CLR PSW. 4
                    CLR PSW. 3
                    CLR FAULT_IND
                    CLR TRIP_SIGNAL
                    LJMP MAIN
;*************************************************************
    INITIALIZE_LCD:         ;lcd initialization sub-routine
                    MOV   A, #38H
                    LCALL  WRITE_CMD
                    MOV   A, #0CH
                    LCALL  WRITE_CMD
                    MOV   A, #06H
                    LCALL  WRITE_CMD
                    RET
;*************************************************************
    WRITE_CMD:      ;sub-routine to write a command to lcd's instruction register
                    CLR    RS          ;rs=0 for selecting command register
                    MOV   P0, A
                    SETB  EN
                    CLR    EN
                    LCALL  LCD_DELAY
                    RET
;*************************************************************
    WRITE_CHAR:             ;sub-routine to write a data to lcd's data register
                    SETB  RS        ;rs=1 for selecting data register
                    MOV   P0, A
                    SETB  EN
                    CLR    EN
                    LCALL  LCD_DELAY
                    RET
```

```
;*************************************************************
        WRITE_STRING:       CLR A
                            MOVC A,@A+DPTR
                            JZ EXIT
                            MOV P0, A
                            SETB RS
                            SETB EN
                            ACALL LCD_DELAY
                            CLR EN
                            INC DPTR
                            SJMP WRITE_STRING
        EXIT:               RET
;*************************************************************
        LCD_DELAY:          ;to generate a delay between consequent lcd write operations
                            MOV   R0, #5H
        L2:                 MOV   R1, #0FFH
        L1:                 DJNZ  R1, L1
                            DJNZ  R0, L2
                            RET
;*************************************************************
        DISPLAY_CURRENT:                ;subroutine to display 'Current:' on lcd
                            MOV   A, #80H
                            LCALL  WRITE_CMD
                            ACALL LCD_DELAY
                            MOV DPTR,#DISPLAY_MAIN_SCREEN
                            ACALL WRITE_STRING
                            ACALL LCD_DELAY
                            RET
;*************************************************************
        DISPLAY:    ;subroutine to display the digits of the measured voltage
                            MOV   A, R6         ;display the digit before decimal point
```

```
                    ADD   A, #30H
                    LCALL WRITE_CHAR
                    MOV   A, #2EH              ;display the decimal point
                    LCALL WRITE_CHAR
                    MOV   A, R5        ;display the digit after the decimal point
                    ADD   A, #30H
                    LCALL WRITE_CHAR
                    MOV   A, R2    ;display the 2ND digit after the decimal point
                    ADD   A, #30H
                    LCALL  WRITE_CHAR
                    MOV   A, #'A'
                    ACALL  WRITE_CHAR
                    ACALL  LCD_DELAY
                    RET
;*****************************************************************
READ_ADC:           ;sub-routine to read from adc
                    SETB  ADC_EOC
                    CLR    ADC_ALE
                    CLR    ADC_SC
                    CLR    ADC_A       ;channel 0 is selected
                    CLR    ADC_B
                    CLR    ADC_C
                    SETB  ADC_ALE    ;latch the address of the channel to adc
                    SETB  ADC_SC                ; start the conversion
                    CLR    ADC_ALE
                    CLR    ADC_SC
WAIT:               JNB ADC_EOC, WAIT       ;conversion complete
                    NOP
                    MOV A,P1
                    NOP
                    SETB ADC_EOC
```

```
                    RET
;*************************************************************
    FIND_MAX_AMPLITUDE:     ;subroutine to find the max amplitude
                    MOV   R7, #0FFH
    REPEAT:         DEC R7
                    LCALL  READ_ADC
     ;find the maximum sampled current amplitude
    NEXT:           MOV   R2, A  ;store the present current amplitude in R2
                    MOV   A, R3  ;load accumulator with previous value
                    SUBB  A, R2        ; subtract current from previous value
            ;       JC MAXIMUM_AMPLITUDE
                    AJMP  CHECK
    MAXIMUM_AMPLITUDE:
                    MOV   A, R2
                    MOV   R3, A
    CHECK:          MOV   A, R7
                    JNZ  REPEAT
                    RET
;*************************************************************
    RMS_FIND:
    ;scale down the input adc value by 5 for DC or  by 7 for AC
                    MOV   A, R3
                    MOV   B, #7D
                    DIV    AB
                    MOV   R4, A
                    MOV R2, B

    ;convert the hex to two separate decimal digits to be displayed
    HEX_TO_DECIMAL:
                    MOV   A, R4
                    MOV   B, #10D          ;divide by 10
```

```
                DIV   AB
                MOV   R5, B      ;r5 contains the value after decimal
                MOV   R6, A      ;r6 contains the value before decimal point
                RET
;************************************************
COMPARE_WITH_SET_VALUE:
                MOV A,R6
                ADD A,R6
                ADD A,R6
                ADD A,R6
                ADD A,R6
                ADD A,R6
                ADD A,R6
                ADD A,R6
                ADD A,R6
                ADD A,R6
                ADD A,R6
                ADD A,R5
                MOV R4,A
                ADD A,R4
                ADD A,R4
                ADD A,R4
                ADD A,R4
                ADD A,R4
                ADD A,R4
                ADD A,R4
                ADD A,R2
                MOV R4,A
                SETB PSW. 4
                MOV A,R3
                ADD A,R3
                ADD A,R3
                ADD A,R3
```

```
ADD A,R3
ADD A,R3
ADD A,R3
ADD A,R3
ADD A,R3
ADD A,R3
ADD A,R5
SETB PSW. 3
MOV R0,A
ADD A,R0
ADD A,R0
ADD A,R0
ADD A,R0
ADD A,R0
ADD A,R0
ADD A,R5
CLR PSW. 3
SETB PSW. 3
MOV R0,A
CLR PSW. 3
CLR PSW. 4
SETB PSW. 4
SETB PSW. 3
MOV A,R0
CLR PSW. 4
CLR PSW. 3
SUBB A,R4
JC CROSSED_SETPOINT

MOV A,#3CH          ; activate second line
ACALL WRITE_CMD
```

```
                    ACALL LCD_DELAY
                    MOV A,#0C0H                 ;jump to second line
                    ACALL WRITE_CMD
                    ACALL LCD_DELAY
                    MOV DPTR,#NOT_CROSSED_SP
                    ACALL WRITE_STRING
                    ACALL LCD_DELAY
                    CLR TRIP_SIGNAL
                    RET
CROSSED_SETPOINT:
                    SETB PSW. 4
                    SETB PSW. 3
COM_CHAR1:          CJNE R2,#1D,COM_CHAR2
                    CLR PSW. 4
                    CLR PSW. 3
                    LCALL TRIP_DELAY_FOR_2SEC
                    LCALL TRIP_COMMAND
                    RET
COM_CHAR2:          CJNE R2,#2D,COM_CHAR3
                    CLR PSW. 4
                    CLR PSW. 3
                    LCALL TRIP_DELAY_FOR_2SEC
                    LCALL TRIP_DELAY_FOR_2SEC
                    LCALL TRIP_COMMAND
                    RET
COM_CHAR3:          CJNE R2,#3D,COM_CHAR4
                    CLR PSW. 4
                    CLR PSW. 3
                    LCALL TRIP_DELAY_FOR_5SEC
                    LCALL TRIP_COMMAND
                    RET
```

```
COM_CHAR4:      CJNE R2,#4D,COM_CHAR5
                CLR PSW. 4
                CLR PSW. 3
                LCALL TRIP_DELAY_FOR_5SEC
                LCALL TRIP_DELAY_FOR_5SEC
                LCALL TRIP_COMMAND
                RET
COM_CHAR5:      CJNE R2,#5D,COM_CHAR6
                CLR PSW. 4
                CLR PSW. 3
                LCALL TRIP_DELAY_FOR_5SEC
                LCALL TRIP_DELAY_FOR_5SEC
                LCALL TRIP_DELAY_FOR_5SEC
                LCALL TRIP_COMMAND
                RET
COM_CHAR6:      CJNE R2,#6D,COM_CHAR1
                CLR PSW. 4
                CLR PSW. 3
                LCALL TRIP_DELAY_FOR_5SEC
                LCALL TRIP_DELAY_FOR_5SEC
                LCALL TRIP_DELAY_FOR_5SEC
                LCALL TRIP_DELAY_FOR_5SEC
                LCALL TRIP_COMMAND
                RET

TRIP_COMMAND:
                SETB TRIP_SIGNAL
                MOV A, #01H
                LCALL  WRITE_CMD
ROTATE:
                MOV   A, #81H
```

```
                    LCALL  WRITE_CMD
                    ACALL LCD_DELAY
                    MOV DPTR,#CROSSED_SP
                    ACALL WRITE_STRING
                    ACALL LCD_DELAY
;second line display
                    MOV A,#3CH              ; activate second line
                    ACALL WRITE_CMD
                    ACALL LCD_DELAY
                    MOV A,#0C0H             ;jump to second line
                    ACALL WRITE_CMD
                    ACALL LCD_DELAY
                    MOV DPTR,#CROSSED_SP2
                    ACALL WRITE_STRING
                    ACALL LCD_DELAY
                    ACALL DISPLAY
                    JNB  LED_RESET, OUT
                    AJMP ROTATE
OUT:                MOV A, #01H
                    LCALL WRITE_CMD
                    ACALL  LCD_DELAY
                    RET
RELAY_SETTING:
                    MOV A, #01H
                    LCALL  WRITE_CMD
                    MOV   A, #81H
                    LCALL  WRITE_CMD
                    ACALL LCD_DELAY
                    MOV DPTR,#RSTG
                    ACALL WRITE_STRING
                    ACALL LCD_DELAY
```

```
MOV DPTR,#RSTG2
ACALL WRITE_STRING
ACALL LCD_DELAY
LCALL DISPLAY_R0
LCALL DISPLAY_2E
LCALL DISPLAY_R5
LCALL DISPLAY_R10
MOV A,#3CH; ACTIVATE SECOND LINE
ACALL WRITE_CMD
ACALL LCD_DELAY
MOV A,#0C1H                ;jump to second line
ACALL WRITE_CMD
ACALL LCD_DELAY
MOV DPTR,#RSTS
ACALL WRITE_STRING
ACALL LCD_DELAY
SETB PSW. 3
SETB PSW. 4
MOV R3,#1D
CLR PSW. 3
CLR PSW. 4
ACALL DI_CHAR
SETB PSW. 3
SETB PSW. 4
MOV R3,#0D
CLR PSW. 3
CLR PSW. 4
LCALL DELAY_FOR_10SEC
LCALL DELAY_FOR_10SEC
JNB RIGHT,NEXT_PAGE
RET
```

```
NEXT_PAGE:      MOV A, #01H
                LCALL  WRITE_CMD
                MOV   A, #81H
                LCALL  WRITE_CMD
                ACALL  LCD_DELAY
                MOV DPTR,#RSTGNEXTPAGE
                ACALL WRITE_STRING
                ACALL LCD_DELAY
NEXT_PAGEUC:    MOV A,#3CH          ; activate second line
                ACALL WRITE_CMD
                ACALL LCD_DELAY
                MOV A,#0C4H         ;jump to second line
                ACALL WRITE_CMD
                ACALL LCD_DELAY
                MOV A,#0EH
                ACALL WRITE_CMD
                ACALL LCD_DELAY
                MOV A,#06H
                ACALL WRITE_CMD
                ACALL LCD_DELAY
                MOV   A, #'I'
                ACALL  WRITE_CHAR
                ACALL  LCD_DELAY
                MOV   A, #':'
                ACALL  WRITE_CHAR
                ACALL  LCD_DELAY
                LCALL DISPLAY_R0
                LCALL DISPLAY_2E
                LCALL DISPLAY_R5
                LCALL DISPLAY_R10
```

```
                    MOV   A, #'A'
                    ACALL  WRITE_CHAR
                    ACALL  LCD_DELAY
                    MOV A,#10H
                    ACALL WRITE_CMD
                    ACALL LCD_DELAY
                    MOV A,#10H
                    ACALL WRITE_CMD
                    ACALL LCD_DELAY

CHECK_LEFT_R1:JB LEFT,OP_R1                          ;left if not pressed
CHECK_LONG_LEFT_R1:  LCALL  DELAY_FOR_10SEC          ;call for wait
                    JB LEFT,LEFT_MOVE_R5             ;left not pressed
                    MOV A, #01H
                    LCALL  WRITE_CMD
                    RET

OP_R1:
CHECK_UP_R1:        JB UP,CHECK_DOWN_R1              ;checking increment port
                    SETB PSW. 4
                     SETB PSW. 3
                    INC R5
                    MOV A,#0C9H                      ;jump to second line
                    ACALL WRITE_CMD
                    ACALL LCD_DELAY
                    ACALL DISPLAY_R10
                    MOV A,#10H
                    ACALL WRITE_CMD
                    ACALL LCD_DELAY
                    CLR PSW. 4
                    CLR PSW. 3
```

```
                        LCALL  DELAY_FOR_10SEC
CHECK_DOWN_R1:          JB DOWN ,CHECK_RIGHT_R1
                        SETB PSW. 4
                        SETB PSW. 3
                        DEC R5
                        MOV A,#0C9H   ;jump to second line
                        ACALL WRITE_CMD
                        ACALL LCD_DELAY
                        ACALL DISPLAY_R10
                        MOV A,#10H
                        ACALL WRITE_CMD
                        ACALL LCD_DELAY
                        CLR PSW. 4
                        CLR PSW. 3

CHECK_RIGHT_R1:         LCALL  DELAY_FOR_10SEC
                        JB RIGHT,CHECK_LEFT
                        LJMP SET_CHAR

CHECK_LEFT:             AJMP CHECK_LEFT_R1
                        RET
LEFT_MOVE_R5:           MOV A,#10H
                        ACALL WRITE_CMD
                        ACALL LCD_DELAY

CHECK_LEFT_R5:          JB LEFT,OP_R5
CHECK_LONG_LEFT_R2:     LCALL  DELAY_FOR_10SEC          ;call for wait
                        JB LEFT,LEFT_MOVE_R0     ;left not pressed
                        MOV A, #01H
                        LCALL WRITE_CMD
                        RET
```

```
OP_R5:
CHECK_UP_R5:        JB UP,CHECK_DOWN_R5        ;checking increment port
                    SETB PSW. 4
                    INC R5
                    MOV A,#0C8H                ;jump to second line
                    ACALL WRITE_CMD
                    ACALL LCD_DELAY
                    ACALL DISPLAY_R5
                    MOV A,#10H
                    ACALL WRITE_CMD
                    ACALL LCD_DELAY
                    CLR PSW. 4
                    LCALL  DELAY_FOR_10SEC
CHECK_DOWN_R5:      JB DOWN ,CHECK_LEFT_R5
                    SETB PSW. 4
                    DEC R5
                    MOV A,#0C8H                ;jump to second line
                    ACALL WRITE_CMD
                    ACALL LCD_DELAY
                    ACALL DISPLAY_R5
                    MOV A,#10H
                    ACALL WRITE_CMD
                    ACALL LCD_DELAY
                    CLR PSW. 4
                    LCALL  DELAY_FOR_10SEC
                    AJMP CHECK_LEFT_R5
                    RET
LEFT_MOVE_R0:       MOV A,#10H
                    ACALL WRITE_CMD
                    ACALL LCD_DELAY
                    MOV A,#10H
```

```
                        ACALL WRITE_CMD
                        ACALL LCD_DELAY
CHECK_LEFT_R0:          JNB RIGHT,JUMP_RIGHT
                        JB LEFT, OP_R0
CHECK_LONG_LEFT_R0:     LCALL DELAY_FOR_10SEC       ;call for wait
                        JB LEFT,OP_R0               ;left not pressed
                        MOV A, #01H
                        LCALL WRITE_CMD
                        RET

    OP_R0:
CHECK_UP_R0:            JB UP,CHECK_DOWN_R0         ;checking increment port
                        SETB PSW. 4
                        INC R3
                        MOV A,#0C6H                 ;jump to second line
                        ACALL WRITE_CMD
                        ACALL LCD_DELAY
                        ACALL DISPLAY_R0
                        MOV A,#10H
                        ACALL WRITE_CMD
                        ACALL LCD_DELAY
                        CLR PSW. 4
                        LCALL  DELAY_FOR_10SEC
CHECK_DOWN_R0:          JB DOWN ,CHECK_LEFT_R0
                        SETB PSW. 4
                        DEC R3
                        MOV A,#0C6H   ;jump to second line
                        ACALL WRITE_CMD
                        ACALL LCD_DELAY
                        ACALL DISPLAY_R0
                        MOV A,#10H
```

```
                        ACALL WRITE_CMD
                        ACALL LCD_DELAY
                        CLR PSW. 4
                        LCALL       DELAY_FOR_10SEC
                        AJMP CHECK_LEFT_R0
                        RET
JUMP_RIGHT:             MOV A,#14H
                        ACALL WRITE_CMD
                        ACALL LCD_DELAY
                        MOV A,#14H
                        ACALL WRITE_CMD
                        ACALL LCD_DELAY
                        MOV A,#14H
                        ACALL WRITE_CMD
                        ACALL LCD_DELAY
                        LJMP CHECK_LEFT_R1
NEXT_PAGEUCC:           AJMP NEXT_PAGEUCC
                        RET
;*****************************************************
SET_CHAR:               MOV A, #01H
                        LCALL  WRITE_CMD
                        MOV   A, #81H
                        LCALL  WRITE_CMD
                        MOV DPTR,#CHAR1
                        ACALL WRITE_STRING
                        ACALL LCD_DELAY
                        MOV A,#3CH          ; activate second line
                        ACALL WRITE_CMD
                        ACALL LCD_DELAY
SET_DEF:                MOV A,#0C4H   ;jump to second line
                        ACALL WRITE_CMD
```

```
                    ACALL LCD_DELAY
                    MOV DPTR,#CHAR_TYPE_DEF
                    ACALL WRITE_STRING
                    ACALL LCD_DELAY
CHECK_LEFT0:        JB LEFT,CHECK_UP0
                    LCALL DELAY_FOR_10SEC
                    JB LEFT,CHECK_UP0
                    RET
CHECK_UP0:          JB UP,CHECK_DOWN0
                    MOV A,#0C4H             ;jump to second line
                    ACALL WRITE_CMD
                    ACALL LCD_DELAY
SET_INV:            MOV DPTR,#CHAR_TYPE_INV
                    ACALL WRITE_STRING
                    ACALL LCD_DELAY
CHECK_DOWN0:        JB DOWN,CHECK_LEFT0
                    MOV A,#0C4H             ;jump to second line
                    ACALL WRITE_CMD
                    ACALL LCD_DELAY
                    MOV DPTR,#CHAR_TYPE_DEF
                    ACALL WRITE_STRING
                    ACALL LCD_DELAY
CHECK_RIGHT0:       JB RIGHT,CHECK_UP2
                    AJMP SET_TIME_DEF_CHAR
CHECK_UP2:          JB UP,CHECK_LEFT2
                    AJMP SET_INV
CHECK_LEFT2:        JB LEFT,CHECK_RIGHT0
                     LCALL DELAY_FOR_10SEC
                    JB LEFT,CHECK_RIGHT0
                    MOV A, #01H
                    LCALL  WRITE_CMD
```

```
                    RET
;*****************************************************
SET_TIME_DEF_CHAR:
                    MOV A, #01H
                    LCALL  WRITE_CMD
                    MOV   A, #81H
                    LCALL  WRITE_CMD
                    MOV DPTR,#CHAR_TIME
                    ACALL WRITE_STRING
                    ACALL LCD_DELAY
                    MOV A,#3CH          ; activate second line
                    ACALL WRITE_CMD
                    ACALL LCD_DELAY
                    MOV A,#0C7H   ;JUMP TO SECOND LINE
                    ACALL WRITE_CMD
                    ACALL LCD_DELAY
    DI_CHAR:
                    SETB PSW. 4
                    SETB PSW. 3
    COM1:           CJNE R2,#1D,COM2
                    CLR PSW. 3
                    CLR PSW. 4
    COM11:
                    MOV A,#0C7H         ;jump to second line
                    ACALL WRITE_CMD
                    ACALL LCD_DELAY
                    MOV DPTR,#D2
                    ACALL WRITE_STRING
                    ACALL LCD_DELAY
                    SETB PSW. 4
                    SETB PSW. 3
```

```
                    MOV R2,#1D
                    CJNE R3,#0D,DWN1
                    CLR PSW. 3
                    CLR PSW. 4
CHECK_LEFT31:       JB LEFT,CHECK_UP31
                    LCALL DELAY_FOR_10SEC
                    JB LEFT,CHECK_UP31
                    MOV A, #01H
                    LCALL   WRITE_CMD
DWN1:               RET
CHECK_UP31:         JB UP,CHECK_LEFT31
                    LCALL  DELAY_FOR_10SEC
                    AJMP COM22
COM2:               CJNE R2,#2D,COM3
                    CLR PSW. 3
                    CLR PSW. 4
COM22:              MOV A,#0C7H             ;jump to second line
                    ACALL WRITE_CMD
                    ACALL LCD_DELAY
                    MOV DPTR,#D4
                    ACALL WRITE_STRING
                    ACALL LCD_DELAY
                    SETB PSW. 4
                    SETB PSW. 3
                    MOV R2,#2D
                    CJNE R3,#0D,DWN2
                    CLR PSW. 3
                    CLR PSW. 4
                    CHECK_LEFT32:JB LEFT,CHECK_UP32
                    LCALL DELAY_FOR_10SEC
                    JB LEFT,CHECK_UP32
```

```
                MOV A, #01H
                LCALL  WRITE_CMD
DWN2:           RET
CHECK_UP32:     JB UP,CHECK_DOWN32
                LCALL DELAY_FOR_10SEC
                AJMP COM33
CHECK_DOWN32:   JB DOWN,CHECK_LEFT32
                LCALL  DELAY_FOR_10SEC
                AJMP COM11
COM3:           CJNE R2,#3D,COM4
                CLR PSW. 3
                CLR PSW. 4
COM33:          MOV A,#0C7H          ;jump to second line
                ACALL WRITE_CMD
                ACALL LCD_DELAY
                MOV DPTR,#D5
                ACALL WRITE_STRING
                ACALL LCD_DELAY
                SETB PSW. 4
                SETB PSW. 3
                MOV R2,#3D
                CJNE R3,#0D,DWN3
                CLR PSW. 3
                CLR PSW. 4
CHECK_LEFT33:   JB LEFT,CHECK_UP33
                LCALL DELAY_FOR_10SEC
                JB LEFT,CHECK_UP33
                MOV A, #01H
                LCALL  WRITE_CMD
DWN3:           RET
CHECK_UP33:     JB UP,CHECK_DOWN33
```

```
                    LCALL  DELAY_FOR_10SEC
                    AJMP COM44
CHECK_DOWN33:  JB DOWN,CHECK_LEFT33
                    LCALL  DELAY_FOR_10SEC
                    AJMP COM22
COM4:               CJNE R2,#4D,COM5
                    CLR PSW. 3
                    CLR PSW. 4
COM44:              MOV A,#0C7H              ;jump to second line
                    ACALL WRITE_CMD
                    ACALL LCD_DELAY
                    MOV DPTR,#D10
                    ACALL WRITE_STRING
                    ACALL LCD_DELAY
                    SETB PSW. 4
                    SETB PSW. 3
                    MOV R2,#4D
                    CJNE R3,#0D,DWN4
                    CLR PSW. 3
                    CLR PSW. 4
CHECK_LEFT34:  JB LEFT,CHECK_UP34
                    LCALL DELAY_FOR_10SEC
                    JB LEFT,CHECK_UP34
                    MOV A, #01H
                    LCALL  WRITE_CMD
DWN4:               RET
CHECK_UP34:    JB UP,CHECK_DOWN34
                    LCALL  DELAY_FOR_10SEC
                    AJMP COM55
CHECK_DOWN34:  JB DOWN,CHECK_LEFT34
                    LCALL  DELAY_FOR_10SEC
```

```
                AJMP COM33
COM5:           CJNE R2,#5D,COM6
                CLR PSW. 3
                CLR PSW. 4
COM55:          MOV A,#0C7H             ;jump to second line
                ACALL WRITE_CMD
                ACALL LCD_DELAY
                MOV DPTR,#D15
                ACALL WRITE_STRING
                ACALL LCD_DELAY
                SETB PSW. 4
                SETB PSW. 3
                MOV R2,#5D
                CJNE R3,#0D,DWN5
                CLR PSW. 3
                CLR PSW. 4
CHECK_LEFT35:   JB LEFT,CHECK_UP35
                LCALL DELAY_FOR_10SEC
                JB LEFT,CHECK_UP35
                MOV A, #01H
                LCALL  WRITE_CMD
DWN5:           RET
CHECK_UP35:     JB UP,CHECK_DOWN35
                LCALL DELAY_FOR_10SEC
                AJMP COM66
CHECK_DOWN35:   JB DOWN,CHECK_LEFT35
                LCALL DELAY_FOR_10SEC
                AJMP COM44
COM6:           CJNE R2,#6D,COM111
                CLR PSW. 3
                CLR PSW. 4
```

```
COM66:          MOV A,#0C7H          ;jump to second line
                ACALL WRITE_CMD
                ACALL LCD_DELAY
                MOV DPTR,#D20
                LCALL WRITE_STRING
                LCALL LCD_DELAY
                SETB PSW. 4
                SETB PSW. 3
                MOV R2,#6D
                CJNE R3,#0D,DWN6
                CLR PSW. 3
                CLR PSW. 4
CHECK_LEFT36:   JB LEFT,CHECK_DOWN36
                LCALL DELAY_FOR_10SEC
                JB LEFT,CHECK_DOWN36
                MOV A, #01H
                LCALL  WRITE_CMD
 DWN6:          RET
CHECK_DOWN36:   JB DOWN,CHECK_LEFT36
                LCALL  DELAY_FOR_10SEC
                LJMP COM55
COM111:         LJMP COM1
                RET
;*********************************************************
SET_VALUE_REGISTER_DISPLAY:
                SETB PSW. 4
                MOV   A, R0          ;display the digit before decimal point
                ADD   A, #30H
                LCALL  WRITE_CHAR
                ACALL  LCD_DELAY
                MOV   A, #2EH
```

```
                ACALL  WRITE_CHAR
                ACALL  LCD_DELAY
                MOV   A, R5          ;display the digit before decimal point
                ADD   A, #30H
                LCALL  WRITE_CHAR
                ACALL  LCD_DELAY
                SETB PSW. 3
                MOV   A, R5          ;display the digit before decimal point
                ADD   A, #30H
                LCALL  WRITE_CHAR
                ACALL  LCD_DELAY
                CLR PSW. 4
                CLR PSW. 3
                RET
;*********************************************************************
    DISPLAY_R10:
                SETB PSW. 4
                SETB PSW. 3
                MOV   A, R5       ;display the 2ND digit after decimal point
                ADD   A, #30H
                LCALL  WRITE_CHAR
                ACALL  LCD_DELAY
                CLR PSW. 4
                CLR PSW. 3
                RET
    DISPLAY_R5:     SETB PSW. 4
                MOV   A, R5          ;display the digit after decimal point
                ADD   A, #30H
                LCALL  WRITE_CHAR
                ACALL  LCD_DELAY
                CLR PSW. 4
```

```
                        RET
DISPLAY_R0:             SETB PSW. 4
                        MOV   A, R3          ;display the digit before decimal point
                        ADD   A, #30H
                        LCALL  WRITE_CHAR
                        ACALL  LCD_DELAY
                        CLR PSW. 4
                        RET
DISPLAY_2E:             MOV   A, #2EH        ;display the decimal point
                        ACALL  WRITE_CHAR
                        ACALL  LCD_DELAY
                        RET
;*********************************************************************
DELAY_FOR_10SEC:        MOV R4,#50D
LL3:                    MOV   R0, #50H
LL2:                    MOV   R1, #55H
LL1:                    DJNZ  R1, LL1
                        DJNZ  R0, LL2
                        DJNZ R4,LL3
                        MOV R4,#0D
                        RET
;*********************************************************************
TRIP_DELAY_FOR_2SEC:    MOV R4,#25D
LL32:                   MOV   R0, #100H
LL22:                   MOV   R1, #183H
LL12:                   DJNZ  R1, LL12
                        SETB FAULT_IND
                        DJNZ  R0, LL22
                        CLR FAULT_IND
                        DJNZ R4,LL32
                        MOV R4,#0D
```

```
                        MOV R0,#0D
                        MOV R1,#0D
                        RET
TRIP_DELAY_FOR_5SEC:
                        MOV R4,#100D
LL35:                   MOV   R0, #199H
LL25:                   MOV   R1, #184H
LL15:                   DJNZ  R1, LL15
                        DJNZ  R0, LL25
                        SETB FAULT_IND
                        DJNZ R4,LL35
                        MOV R4,#0D
                        MOV R0,#0D
                        MOV R1,#0D
                        RET
;*************************************************************
        RSTG:           DB "CSET. :",0
        RSTG2:          DB "I:",0
        RSTS:           DB "TSET. :",0
        RSTS2:          DB "T:",0
        RSTGNEXTPAGE:   DB "OVER CURRENT:",0
        CROSSED_SP:     DB "Fault in Line ",0
        CROSSED_SP2:    DB "Current I:",0
        NOT_CROSSED_SP: DB "No Fault in Line ",0
        DISPLAY_MAIN_SCREEN:DB "Current:",0
        CHAR1:          DB "SET RELAR CHAR. ",0
        CHAR_TYPE_INV:  DB "Inverse",0
        CHAR_TYPE_DEF:  DB "Definite",0
        CHAR_TIME:      DB "SET DEF.  TIME",0
        D2:             DB "2 Sec",0
        D4:             DB "4 Sec",0
```

```
        D5:                     DB "5 Sec",0
        D10:                    DB "10 Sec",0
        D15:                    DB "15 Sec",0
        D20:                    DB "20 Sec",0
;*************************************************************
        MAIN:                   MOV  R3, #0H            ;default max amplitude
                                MOV  P1, #0FFH      ;set p1 input port
                                MOV  P0, #0H            ;set p0 output port
                                LCALL INITIALIZE_LCD
                                LCALL  DISPLAY_CURRENT
                                LCALL  FIND_MAX_AMPLITUDE
                                LCALL  RMS_FIND
                                LCALL  DISPLAY
                                LCALL  COMPARE_WITH_SET_VALUE
                                JB RIGHT,NEXT_LINE      ;right key not pressed
                                LCALL RELAY_SETTING
        NEXT_LINE:              MOV A,#80H
                                LCALL  WRITE_CMD
                                AJMP   MAIN
                                END
```